IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Lwo	**Conf. No.:**	7441
Serial No.:	10/783,002	**Art Unit:**	2191
Filing Date:	02/20/2004	**Examiner:**	Chen, Qing
Title:	COMPUTER-IMPLEMENTED METHOD, SYSTEM AND PROGRAM PRODUCT FOR COMPARING APPLICATION PROGRAM INTERFACES (APIs) BETWEEN JAVA BYTE CODE RELEASES	**Docket No.:**	RSW920030291US1 (IBMR-0067)

Mail Stop Appeal Brief- Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

BRIEF OF APPELLANTS

This is an appeal from the Final Rejection dated April 16, 2009, rejecting claims 1-27.

This Brief is accompanied by the requisite fee set forth in 37 C.F.R. 1.17 (c).

REAL PARTY IN INTEREST

International Business Machines Corporation is the real party in interest.

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF CLAIMS

As filed, this case included claims 1-27. Claims 1-27 remain pending. Claims 1-27 stand rejected and form the basis of this appeal.

STATUS OF AMENDMENTS

A response was submitted in response to the Final Rejection filed by the Office on April 16, 2009; however no amendment to the claims was included therein.

SUMMARY OF THE CLAIMED SUBJECT MATTER

Under the present invention, source input corresponding to a first release of Java byte code and target input corresponding to a second release of the Java byte code is received. The input is transformed into a first list containing class names associated with the first release and a second list containing class names associated with the second release. Thereafter, any classes corresponding to class names that appear on both lists (e.g., matching class names) are loaded. The methods within the matching classes are then compared to determine if any of the APIs have been modified between the two releases. After the comparison, the matching class names are removed from the lists. Any class names remaining on the first list represent APIs that have been removed for the second release, while any class names remaining on the second list represent APIs that have been added for the second release.

Claim 1 claims a computer-implemented method for comparing Application Program Interfaces (APIs) between byte code releases, comprising: receiving source input corresponding to a first release of byte code and target input corresponding to a second release of the byte code (see e.g., para. 0027; Fig. 1, items 14A, 16A, 34, 56, 58); transforming the source input into a

first list that contains class names associated with the first release of byte code, and the target input into a second list containing class names associated with the second release of the byte code (see e.g., para. 0028; Fig. 1, items 36, 56, 58, 62A, 62B); finding matching class names between the first list and the second list, and loading classes corresponding to the matching class names (see e.g., para. 0028; Fig. 1, items 38, 52, 62A, 62B); comparing the loaded classes to identify APIs that have been modified between the first release of byte code and the second release of the byte code , wherein an API has not been modified in case that it maintains a same name, parameter order, parameter types and return types in both the first release of the byte code and the second release of the byte code (see e.g., para. 0029; Fig. 1, items 14B, 16B, 42, 62C); and removing the matching class names from the first list and the second list after the comparing (see e.g., para. 0029; Fig. 1, items 44, 62A, 62B), wherein any class names remaining in the first list represent APIs that have been removed for the second release of the byte code, and wherein any class names remaining in the second list represent APIs that have been added for the second release of the byte code (see e.g., para. 0029; Fig. 1, items 62A, 62B).

Claim 10 claims a system for comparing Application Program Interfaces (APIs) between byte code releases, comprising: a computer device; an input system for receiving source input corresponding to a first release of byte code and target input corresponding to a second release of the byte code (see e.g., para. 0027; Fig. 1, items 14A, 16A, 34, 56, 58); a transformation system for transforming the source input into a first list that contains class names associated with the first release of byte code, and the target input into a second list containing class names associated with the second release of the byte code (see e.g., para. 0028; Fig. 1, items 36, 56, 58, 62A, 62B); a class matching system for finding matching class names between the first list and the second list (see e.g., para. 0028; Fig. 1, items 38, 52, 62A, 62B); a class comparison system for

comparing classes corresponding to the matching class names to identify APIs that have been modified between the first release of byte code and the second release of the byte code, wherein an API has not been modified in case that it maintains a same name, parameter order, parameter types and return types in both the first release of the byte code and the second release of the byte code (see e.g., para. 0029; Fig. 1, items 14B, 16B, 42, 62C); and a removal system for removing the matching class names from the first list and the second list after the comparison (see e.g., para. 0029; Fig. 1, items 44, 62A, 62B), wherein any class names remaining in the first list represent APIs that have been removed for the second release of the byte code, and wherein any class names remaining in the second list represent APIs that have been added for the second release of the byte code (see e.g., para. 0029; Fig. 1, items 62A, 62B).

Claim 19 claims a program product stored on a recordable medium for comparing Application Program Interfaces (APIs) between byte code releases, which when executed, comprises: program code for receiving source input corresponding to a first release of byte code and target input corresponding to a second release of the byte code (see e.g., para. 0027; Fig. 1, items 14A, 16A, 34, 56, 58); program code for transforming the source input into a first list that contains class names associated with the first release of byte code, and the target input into a second list containing class names associated with the second release of the byte code (see e.g., para. 0028; Fig. 1, items 36, 56, 58, 62A, 62B); program code for finding matching class names between the first list and the second list (see e.g., para. 0028; Fig. 1, items 38, 52, 62A, 62B); program code for comparing classes corresponding to the matching class names to identify APIs that have been modified between the first release of byte code and the second release of the byte code, wherein an API has not been modified in case that it maintains a same name, parameter order, parameter types and return types in both the first release of the byte code and the second

release of the byte code (see e.g., para. 0029; Fig. 1, items 14B, 16B, 42, 62C); and program code for removing the matching class names from the first list and the second list after the comparison (see e.g., para. 0029; Fig. 1, items 44, 62A, 62B), wherein any class names remaining in the first list represent APIs that have been removed for the second release of the byte code, and wherein any class names remaining in the second list represent APIs that have been added for the second release of the byte code (see e.g., para. 0029; Fig. 1, items 62A, 62B).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 3-6 and 9 stand rejected under 35 U.S.C. §112, second paragraph, as being indefinite.
2. Claims 1-6, 8, 10-15, 17, 19-24 and 26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Schwabe (U.S. Patent No. 6,986,132), hereafter "Schwabe," in view of Judge *et al.* (U.S. Patent No. 6,430,564), hereafter "Judge."
3. Claims 7, 9, 16, 18, 25 and 27 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Schwabe in view of Judge and further in view of Connelly *et al.* (U.S. Patent No. 6,385,722), hereafter "Connelly."

ARGUMENT

1. REJECTION OF CLAIMS 3-6 AND 9 UNDER 35 U.S.C. §112

The Examiner has asserted that claims 3-6 and 9 are indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. Specifically, the Examiner objects to use of the word "JAVA" in the claims. Applicant respectfully submits that "...The presence of a trademark or trade name in a claim is not, *per se*, improper under 35 U.S.C. §112, second paragraph..." MPEP 2173.05(u). To this extent, the MPEP does not strictly forbid the use of trademarks in the claims, but rather only those which are not "...sufficiently precise and definite." MPEP §608.01(v). Applicant asserts that the word JAVA, in the context of the claimed invention, refers to an environment within which the invention functions and/or a framework that defines the structure of the constructs of the invention, and not simply a source of goods. To this extent, the term JAVA has a definite meaning, and its use in the claims is permitted. Accordingly, Applicant requests that the rejection be withdrawn.

2. REJECTION OF CLAIMS 1-6, 8, 10-15, 17, 19-24 AND 26 UNDER §103(a) OVER SCHWABE IN VIEW OF JUDGE

Appellants respectfully submit that the Examiner's rejections of the claims under 35 U.S.C. §103(a) are defective.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify a reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim

limitations. Appellants respectfully submit that the Schwabe and Judge references, taken alone or in combination, fail to meet each of the three basic criteria required to establish a *prima facie* case of obviousness. As such, the rejection under 35 U.S.C. §103(a) is defective.

In the above referenced Final Office Action, the Examiner alleges that the cited references teach or suggest comparing the loaded classes to identify APIs that have been modified between the first release of byte code and the second release of the byte code, wherein an API has not been modified in case that it maintains a same name, parameter order, parameter types and return types in both the first release of the byte code and the second release of the byte code. In the argument to the contrary, the Examiner cites a passage of Schwabe that describes comparison of package attributes. Col. 25, lines 50-53; col. 26, lines 1-10. However, a closer reading of the text surrounding the passages cited by the Office clearly shows that these package attributes are part of the content of API definition files. See e.g., col. 25, lines 37-43. To this extent, the parameters that are compared are in API definition files. As such, Schwabe never compares loaded classes, but only API definition files which are separate from these classes. In furtherance of this, Schwabe expressly teaches that

> ...verification does not continue beyond an API definition file. This differs from typical verification methods that continue the verification process into an implementation of the API definition file. Col. 14, lines 10-13.

To this extent, Schwabe teaches against the loading of classes based on results from an API definition file verification in its verification process. Furthermore, neither of the references teaches or suggests the exact comparison of the claimed invention, including, *inter alia*, order of parameters and both parameter types and return types.

In the above referenced Final Office Action, the Examiner alleges that the cited references teach or suggest finding matching class names between the first list and the second

list, and loading classes corresponding to the matching class names. However, the passages in Schwabe that the Examiner equates to this limitation of the claimed invention refer to different functions of Schwabe that are completely unrelated. The first passage of Schwabe cited by the Office mentions comparing the set of classes and interfaces in an old API definition file with those in a new API definition file. The second passage references loading of class files. However, the passages in Schwabe cited by the Examiner do not teach that the class files that are loaded are class files that are derived from the matching. In fact, Schwabe does not even indicate that the class files that are loaded are taken from its API definition files. Rather, as stated elsewhere herein, Schwabe teaches against comparing of loaded classes, but only compares the definition files. This is in contrast to the claimed invention in which the classes that are loaded correspond to the matching class names between the first list and the second list. For the above reasons, the separate comparing and loading of Schwabe does not teach or suggest the loading based on the matching of the claimed invention. Judge does not cure this deficiency.

3. REJECTION OF CLAIMS 7, 9, 16, 18, 25 AND 27 UNDER 35 U.S.C. §103(a) OVER SCHWABE IN VIEW OF JUDGE AND FURTHER IN VIEW OF CONNELLY

Appellants incorporate the above enumerated arguments. Connelly fails to cure this deficiency.

CONCLUSION

In summary, Appellants submit that claims 1-27 are allowable because the cited references, taken alone or in combination, fail to meet each of the three basic criteria required to establish a *prima facie* case of obviousness.

Respectfully submitted,

/Hunter E. Webb/

Hunter E. Webb
Reg. No.: 54,593

Date: September 16, 2009
Hoffman Warnick LLC
75 State Street, 14th Floor
Albany, New York 12207
(518) 449-0044
(518) 449-0047 (fax)

CLAIMS APPENDIX

Claim Listing:

1. A computer-implemented method for comparing Application Program Interfaces (APIs) between byte code releases, comprising:

receiving source input corresponding to a first release of byte code and target input corresponding to a second release of the byte code;

transforming the source input into a first list that contains class names associated with the first release of byte code, and the target input into a second list containing class names associated with the second release of the byte code;

finding matching class names between the first list and the second list, and loading classes corresponding to the matching class names;

comparing the loaded classes to identify APIs that have been modified between the first release of byte code and the second release of the byte code, wherein an API has not been modified in case that it maintains a same name, parameter order, parameter types and return types in both the first release of the byte code and the second release of the byte code; and

removing the matching class names from the first list and the second list after the comparing, wherein any class names remaining in the first list represent APIs that have been removed for the second release of the byte code, and wherein any class names remaining in the second list represent APIs that have been added for the second release of the byte code.

2. The computer-implemented method of claim 1, further comprising outputting a report identifying at least one of the APIs that have been modified, the APIs that have been removed and the APIs that have been added.

3. The computer-implemented method of claim 1, wherein the loading step comprises loading at least one Java class of the first release of byte code and at least one class of the second release of the byte code.

4. The computer-implemented method of claim 3, further comprising listing methods of the at least one class of the first release of byte code in the first list, and listing methods of the at least one class of the second release of the byte code in the second list.

5. The computer-implemented method of claim 4, wherein the comparing step comprises comparing the methods in the first list to the methods in the second list to identify APIs that have been modified between the first release of byte code and the second release of the byte code.

6. The computer-implemented method of claim 5, wherein the removing step comprises removing, from the first list and the second list, any methods in the first list that are identical to methods in the second list based on the comparison, wherein any methods remaining in the first list after the removing represent APIs that have been removed for the second release of the byte code, and wherein any methods remaining in the second list after the removing represent APIs that have been added for the second release of the byte code.

7. The computer-implemented method of claim 1, wherein the source input and the target input comprise JAR files.

8. The computer-implemented method of claim 1, wherein the source input and the target input comprise a list of classes.

9. The computer-implemented method of claim 1, further comprising inputting class paths common to the first release of Java byte code and the second release of the byte code.

10. A system for comparing Application Program Interfaces (APIs) between byte code releases, comprising:

a computer device;

an input system for receiving source input corresponding to a first release of byte code and target input corresponding to a second release of the byte code;

a transformation system for transforming the source input into a first list that contains class names associated with the first release of byte code, and the target input into a second list containing class names associated with the second release of the byte code;

a class matching system for finding matching class names between the first list and the second list;

a class comparison system for comparing classes corresponding to the matching class names to identify APIs that have been modified between the first release of byte code and the second release of the byte code, wherein an API has not been modified in case that it maintains a same name, parameter order, parameter types and return types in both the first release of the byte code and the second release of the byte code; and

a removal system for removing the matching class names from the first list and the second list after the comparison, wherein any class names remaining in the first list represent

APIs that have been removed for the second release of the byte code, and wherein any class names remaining in the second list represent APIs that have been added for the second release of the byte code.

11. The system of claim 10, further comprising a report generation system for generating a report identifying at least one of the APIs that have been modified, the APIs that have been removed and the APIs that have been added.

12. The system of claim 10, further comprising a class loader for loading at least one class of the first release of byte code and at least one class of the second release of the byte code.

13. The system of claim 12, further comprising a method listing system for listing methods of the at least one class of the first release of byte code in the first list, and for listing methods of the at least one class of the second release of the byte code in the second list.

14. The system of claim 13, wherein the class comparison system compares the methods in the first list to the methods in the second list to identify APIs that have been modified between the first release of byte code and the second release of the byte code.

15. The system of claim 14, wherein the removal system removes, from the first list and the second list, any methods in the first list that are identical to methods in the second list based on the comparison, wherein any methods remaining in the first list after the removing represent APIs that have been removed for the second release of the byte code, and wherein any methods

remaining in the second list after the removing represent APIs that have been added for the second release of the byte code.

16. The system of claim 10, wherein the source input and the target input comprise JAR files.

17. The system of claim 10, wherein the source input and the target input comprise a list of classes.

18. The system of claim 10, wherein the input system further receives class paths common to the first release of byte code and the second release of the byte code.

19. A program product stored on a recordable medium for comparing Application Program Interfaces (APIs) between byte code releases, which when executed, comprises:

program code for receiving source input corresponding to a first release of byte code and target input corresponding to a second release of the byte code;

program code for transforming the source input into a first list that contains class names associated with the first release of byte code, and the target input into a second list containing class names associated with the second release of the byte code;

program code for finding matching class names between the first list and the second list;

program code for comparing classes corresponding to the matching class names to identify APIs that have been modified between the first release of byte code and the second release of the byte code, wherein an API has not been modified in case that it maintains a same

name, parameter order, parameter types and return types in both the first release of the byte code and the second release of the byte code; and

program code for removing the matching class names from the first list and the second list after the comparison, wherein any class names remaining in the first list represent APIs that have been removed for the second release of the byte code, and wherein any class names remaining in the second list represent APIs that have been added for the second release of the byte code.

20. The program product of claim 19, further comprising program code for generating a report identifying at least one of the APIs that have been modified, the APIs that have been removed and the APIs that have been added.

21. The program product of claim 19, further comprising program code for loading at least one class of the first release of byte code and at least one class of the second release of the byte code.

22. The program product of claim 21, further comprising program code for listing methods of the at least one class of the first release of byte code in the first list, and for listing methods of the at least one class of the second release of the byte code in the second list.

23. The program product of claim 22, wherein the program code for comparing compares the methods in the first list to the methods in the second list to identify APIs that have been modified between the first release of byte code and the second release of the byte code.

24. The program product of claim 23, wherein the program code for removing removes, from the first list and the second list, any methods in the first list that are identical to methods in the second list based on the comparison, wherein any methods remaining in the first list after the removing represent APIs that have been removed for the second release of the byte code, and wherein any methods remaining in the second list after the removing represent APIs that have been added for the second release of the byte code.

25. The program product of claim 19, wherein the source input and the target input comprise JAR files.

26. The program product of claim 19, wherein the source input and the target input comprise a list of classes.

27. The program product of claim 19, wherein the program code for receiving further receives class paths common to the first release of byte code and the second release of the byte code.

EVIDENCE APPENDIX

No evidence is entered and relied upon in the appeal.

RELATED PROCEEDINGS APPENDIX

No decisions rendered by a court or the Board in any proceeding are identified in the related appeals and interferences section.